UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

DTIC FILE COPY

2

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
AD-A217 115	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
NMSU-ECE-88-003	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARO 25173.22-EC
6a. NAME OF PERFORMING ORGANIZATION New Mexico State University / 6b. OFFICE SYMBOL (If applicable) PARL	7a. NAME OF MONITORING ORGANIZATION U. S. Army Research Office
6c. ADDRESS (City, State, and ZIP Code) Las Cruces, NM 88003	7b. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211
8a. NAME OF FUNDING/SPONSORING ORGANIZATION U. S. Army Research Office / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DAAL03-87-K-0106
8c. ADDRESS (City, State, and ZIP Code) P. O. Box 12211 Research Triangle Park, NC 27709-2211	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A Prototype Virtual Port Memory Multiprocessor

12. PERSONAL AUTHOR(S)
Eric E. Johnson

13a. TYPE OF REPORT	13b. TIME COVERED FROM ____ TO ____	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT

16. SUPPLEMENTARY NOTATION The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A Small Bus-based Prototype Virtual Port Memory Machine is Described.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

A PROTOTYPE VIRTUAL PORT MEMORY MULTIPROCESSOR

ERIC E. JOHNSON

NMSU-ECE-88-003
May 1988

A PROTOTYPE VIRTUAL PORT MEMORY MULTIPROCESSOR

Eric E. Johnson

I. VIRTUAL PORT MEMORY ARCHITECTURES

The virtual port memory class of multiprocessor architectures was defined in [JOHNS87, JOHNS88] as follows:

> **Definition:** A **virtual port memory multiprocessor architecture** provides each process of a computation or concurrent system with a private virtual address space and pass by value message passing semantics, based upon an underlying hardware structure consisting of a global memory, equally accessible to all processors, and a pass by reference message network.

The techniques employed to implement a virtual port memory architecture may include hardware, firmware, and software elements, with many architectural variations possible to accommodate a wide range of applications.

The programmer's model of such a system is shown in Figure 1. The principal features of this view of the architecture are the multiple processors, each with a private memory, and a high-bandwidth interprocessor message channel (or network) used to pass objects by value.

One approach to implementing this architecture is shown in Figure 2. Each processor is connected to a private port through which to access the shared memory, similar to some shared memory designs; a significant difference is that addresses presented at the ports are virtual addresses, and each port constitutes a distinct virtual address space.

The topic of this paper is a system level description of a prototype implementation of this type of multiprocessor architecture. Work is currently underway at New Mexico State University to construct a 4 PE prototype and to develop the necessary system software to evaluate the usefulness of this architecture. This machine will also serve as a test bed for related architecture and parallel programming research, including investigation of novel processor architectures, development of a model for program reference behavior, and evaluation of cache prefetching techniques.

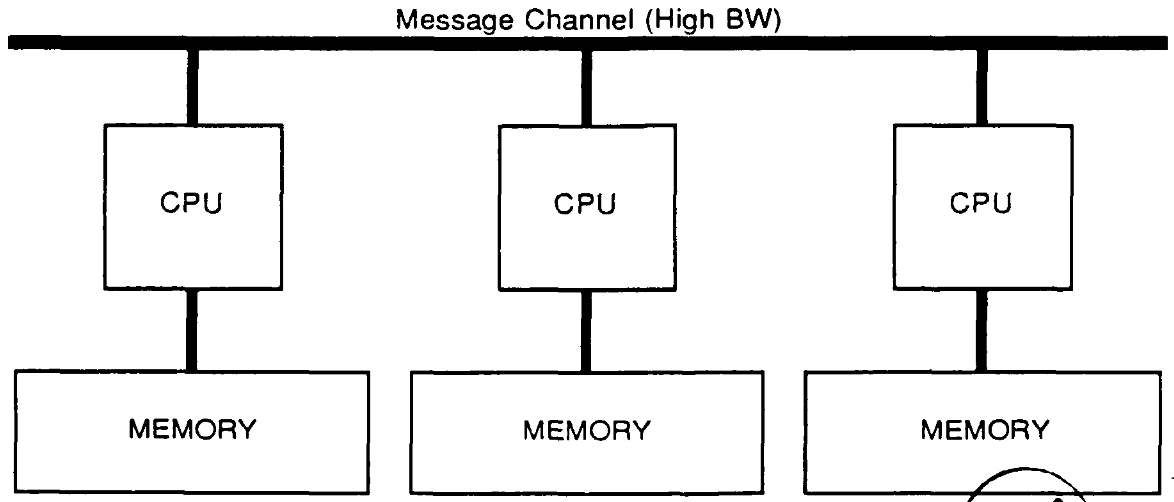


Figure 1: Virtual Port Memory Programmer's Model

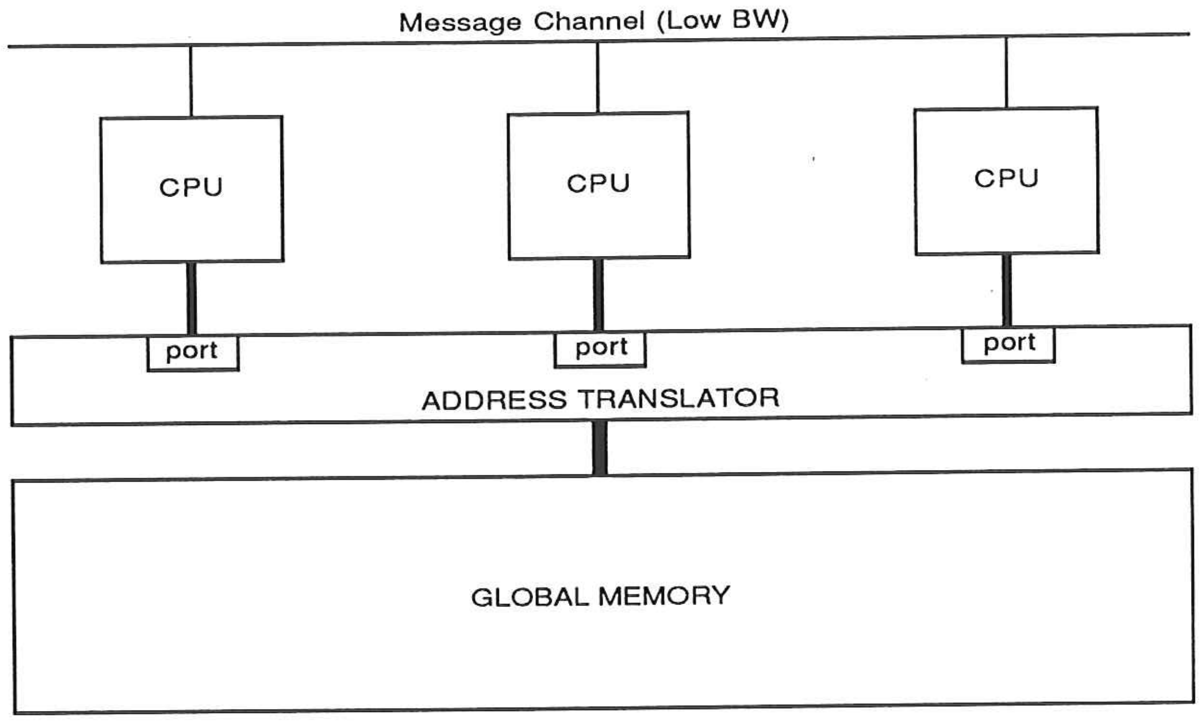


Figure 2: A Virtual Port Memory Implementation Model

II. THE PROTOTYPE ARCHITECTURE

Figure 3 shows a block diagram of a simple bus-based virtual port memory implementation. Execution of application and most system processes takes place on the processing elements (PEs). User interface processors (UIPs) are connected to user workstations and execute shell processes, while input/output controllers (IOCs) manage other I/O. The three processor types share the global memory, and pass interprocess messages among themselves over the interprocessor message bus (IMB). Requests for use of the global memory are sent from the processors to the address translator (ATran) over a transaction request bus (TRB); translated requests are queued in the memory controller for action by the appropriate bank of the interleaved memory, and data transfer takes place over the data transfer bus (DTB). Page copy hardware is contained within the memory banks for fast and efficient memory-to-memory DMA; this minimizes the performance impact of copy-on-write faults, and is also useful for general data movement.

The following sections discuss details of the prototype machine on a module by module basis. Section III describes the interconnection subsystem, including the buses and TRB arbiter. Section IV addresses the global memory subsystem, including the bulk DRAM with its cycle and refresh controllers, the page copy mechanism, and the address translator. Section V discusses the I/O subsystem implemented in the prototype, while section VI describes the structure of the prototype processing element. The paper concludes with a discussion of a prototype operating system in section VII.

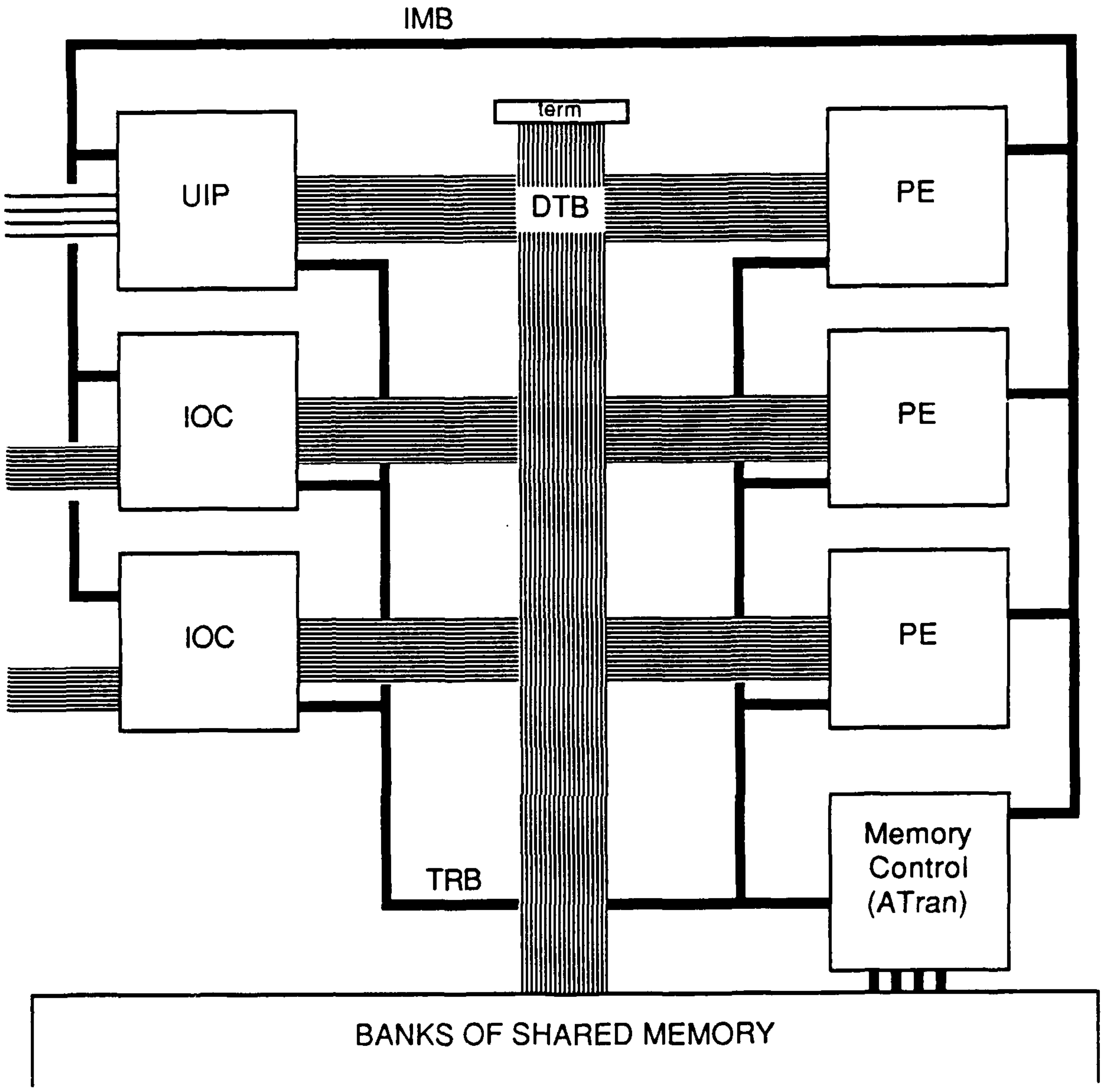


Figure 3: A Bus-Based Virtual Port Memory Prototype

III. INTERCONNECTION SUBSYSTEM

The interconnection subsystem for the prototype machine forms the foundation on which the rest of the machine is built. The design parameters of this prototype interconnection subsystem were chosen to support a total of 15 processor modules (PEs, IOCs, and UIPs) and one memory controller.

The electrical and mechanical specifications for the buses and modules are quite similar to those for the VMEbus [MOT85], except that the buses operate synchronously (16 MHz), and the TTL-level drivers and receivers specified for the VMEbus may be replaced by higher-performance trapezoidal waveform drivers and receivers.

The signal and power circuits will enter the processor and memory controller modules through two 96-pin DIN connectors. (The modules will be standard 6U by 160 mm Eurocards.) These circuits may be grouped as follows:

BUS	CIRCUITS	DIRECTION	LINES
Transaction Request Bus (TRB)	Address	P » M	32
	Port #	P » M	5
	Trans. data	P » M	32
	Control	P » M	5
	Arbitration	P «» Arb.	2
Data Transfer Bus (DTB)	Data	Bidirectional	32
	Port #	M » P	5
	Control	M » P	5
Interprocessor Message Bus (IMB)	Msg. data	P » P	32
	Control	P » P	6
	Arbitration	P «» Arb.	2
Utility Bus	Slot #	» P	5
	Reset	» P	1
	SysClock	» P	1
	SysQClock	» P	1
	+5 v	» P	12
	Return	« P	12

These buses operate as follows: when a processor module (e.g., the PE cache controller) requires a global memory cycle, it must first request and be granted access to the Transaction Request Bus (TRB). The processor module then places its port # (obtained from the utility bus) along with the (virtual) address and any data for the transaction on the TRB for one clock cycle; these signals are received by the memory controller, which immediately begins transaction processing, usually by translating the virtual address to a physical address (one clock cycle).

In the case of a simple read or write cycle, the translated address, port # and cycle type are then queued at the appropriate memory bank (see next section); when the memory bank is ready to perform the cycle, the DTB is connected to the appropriate memory bank, and the data transfer is initiated by the DTB controller by driving the port # onto the bus and asserting appropriate control signals. The cycle runs synchronously, because all devices connected to the DTB operate at full DTB speed.

The IMB is a synchronous processor-processor channel. Control of the IMB is allocated by a dedicated IMB arbiter on a message-by-message basis (messages may take more than one cycle). Processors (and the memory controller) monitor the process IDs of messages on the bus, and queue messages addressed to resident processes in a receive FIFO.

IV. MEMORY SUBSYSTEM

The prototype memory subsystem consists of several cooperating entities: the address translator (ATran), several interleaved banks of dynamic RAM with associated cycle controllers, a refresh timing unit, a page copy unit, a data network, and the DTB controller (see below).

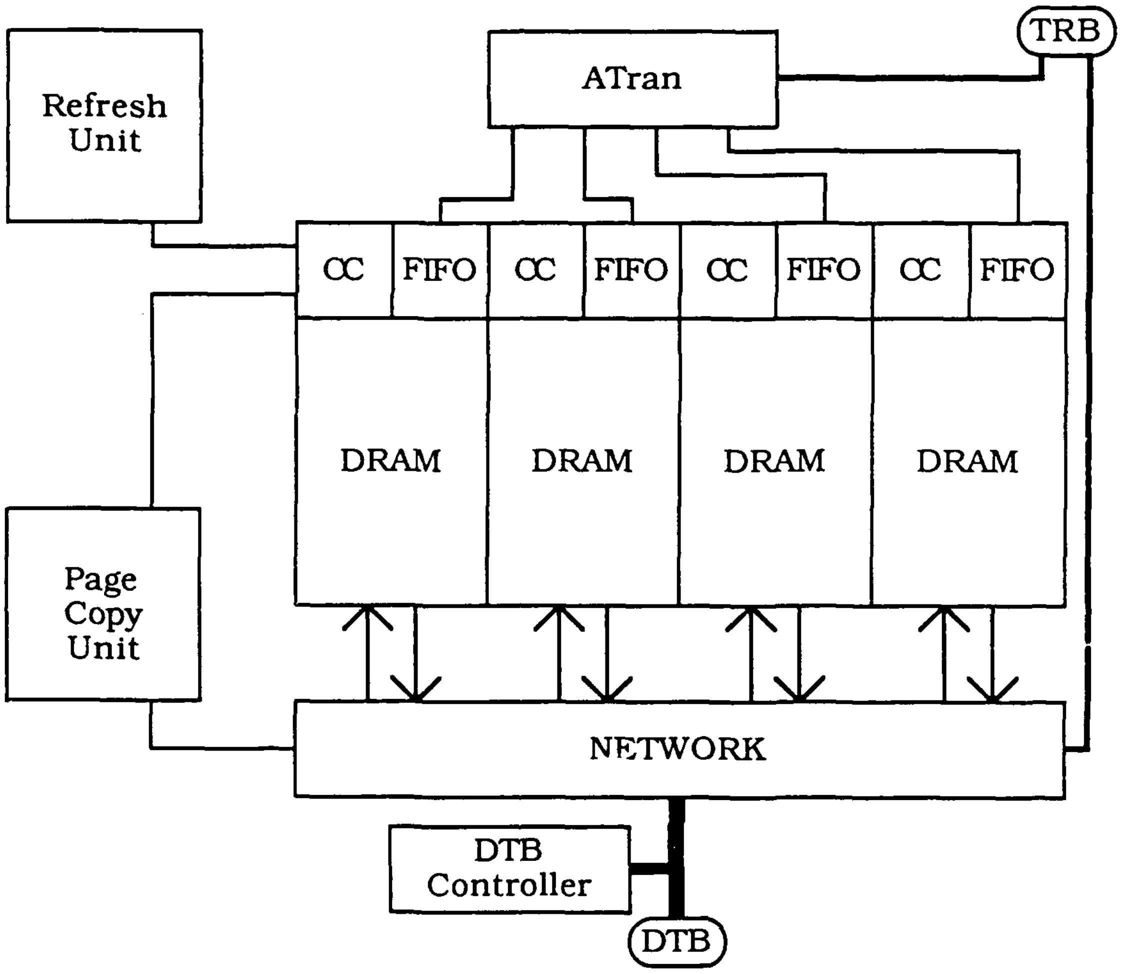


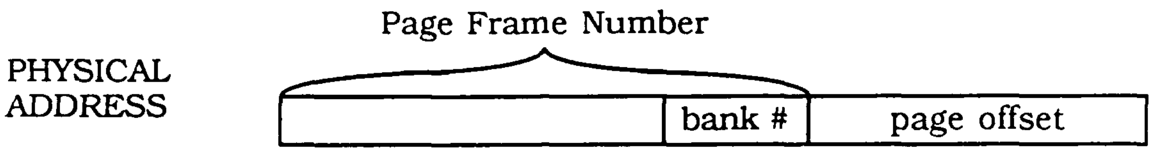
Figure 4: Prototype Global Memory Subsystem

The central components are, of course, the memory modules themselves. In order to match the performance of the DTB, multiple interleaved banks of DRAM are employed. Each is controlled by a dedicated cycle controller (CC in Figure 4), which sequences the control, data, and address signals to the DRAMs.

Also associated with each bank of DRAM is a request queue, implemented as FIFO buffers. These queues contain requests for transactions or page copy cycles. Each entry contains a code for the requested cycle type (including direction of data movement), the physical address of the affected memory operand(s), and may also contain transaction data.

The refresh unit is simply a timer which signals the need for a refresh cycle to the DRAM cycle controllers at appropriate intervals. Refresh requests are accommodated through use of "hidden refresh," which adds a CAS-before-RAS refrech cycle to the end of a normal read or write cycle; if no cycle is in progress when the refresh request arrives, the cycle controller launches a CAS-before-RAS refresh cycle. Both refresh methods employ refresh row address counters internal to the DRAMs.

The address translator (ATran) is the "front end" of the memory subsystem, as seen by the processors. It translates the system virtual addresses arriving in transaction requests to the physical addresses used within the memory subsystem. A full-scale virtual port memory multiprocessor would employ an address translation cache to store the most recently used virtual-to-physical address mappings; this reduced-size prototype will support a smaller system virtual address space, and can therefore store all virtual-physical pairs in a simple RAM within the ATran. Translated requests are passed from the ATran to the request queue of the DRAM bank selected by low-order bits of the page frame number in the physical address:



The page copy unit receives requests to perform memory-to-memory copy operations from other processors via the IMB. The data movement involved in these page copy operations takes place entirely within the memory subsystem. The page copy unit places requests for corresponding read and write memory cycles at the head of the request queues of the affected memory banks; data is then transferred between banks (or from one bank back to the same bank) through the memory data network. Page copy requests could severely reduce the throughput rate of external (TRB) requests if no flow control were imposed; the page copy unit therefore generates its pre-emptive requests only after affected memory banks have had the opportunity to perform an intervening externally requested cycle.

The data network mentioned above contains more than simple data buses to connect the DRAM data inputs and outputs to the DTB. This network has the ability to route internal page copy traffic between banks, to latch data read from a bank so that it may be copied back to that bank, and to perform arithmetic and logical operations on operands from transaction requests and the memory banks.

Finally, the DTB controller works with the cycle controllers of the DRAM banks to manage the flow of data from the memory banks to the processors over the DTB.

V. I/O SUBSYSTEM

The function of the I/O subsystem in the prototype virtual port memory machine is to perform automatic transfer of data between the global memory and external I/O devices. The initial IOC for the prototype machine (Figure 5) will interface the system to an external VMEbus, and will operate as an intelligent I/O processor, executing programs from on-board ROM in response to messages received over the IMB.

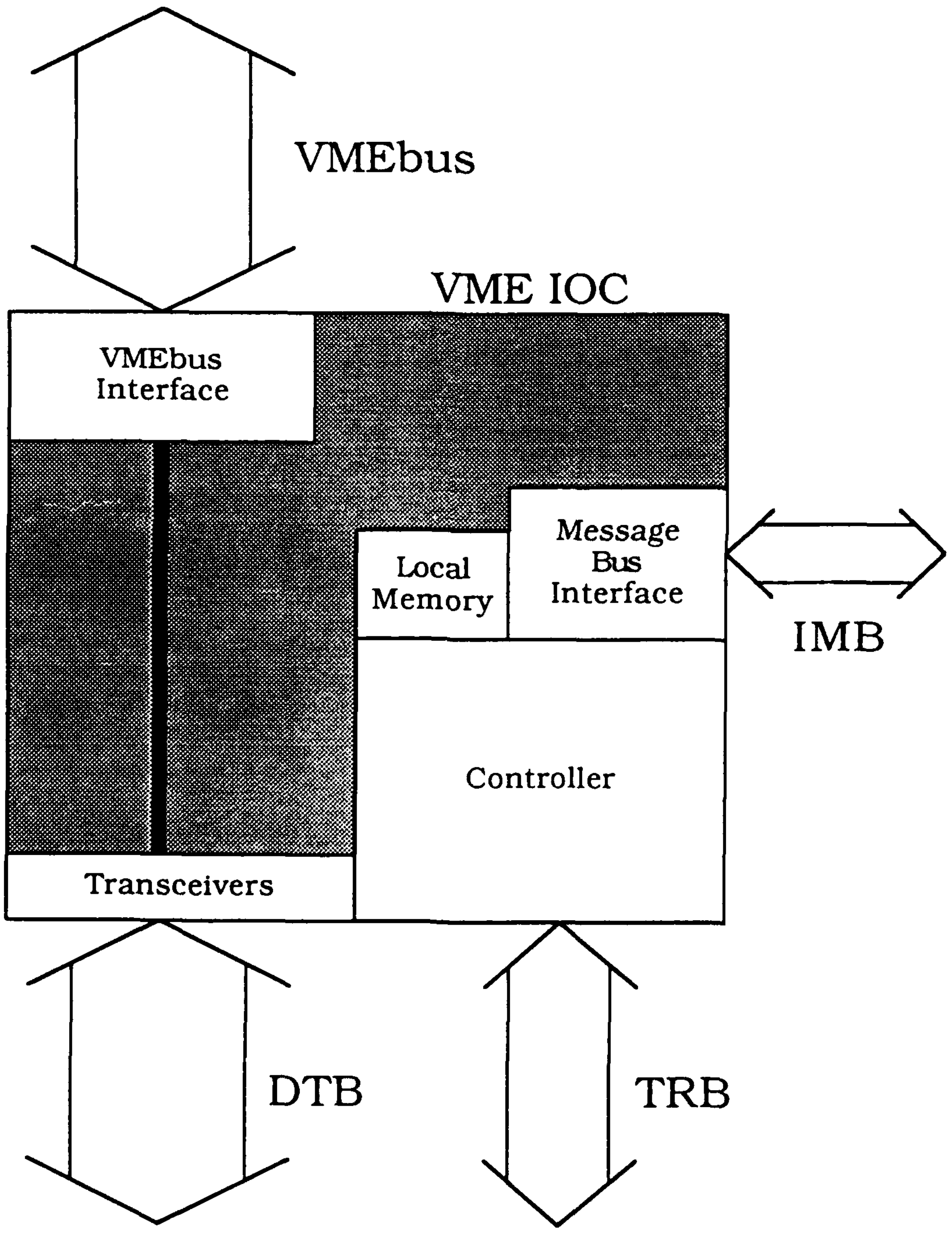


Figure 5: Prototype VMEbus IOC

VI. PROTOYPE PROCESSING ELEMENT

Figure 6 shows a block diagram of the prototype PE. Due to the freedom from cache coherence concerns in a virtual port memory architecture, each PE may make full use of a private cache memory; indeed, caches may be made quite large to minimize traffic on the system bus. The cache controller on the PE generates TRB requests upon cache misses and write-backs, and may also initiate prefetch requests to minimize cache misses seen by the execution unit (EU).

The message unit queues messages to be sent over the IMB, as well as messages so received, and may be enabled to interrupt the EU upon receipt of a message.

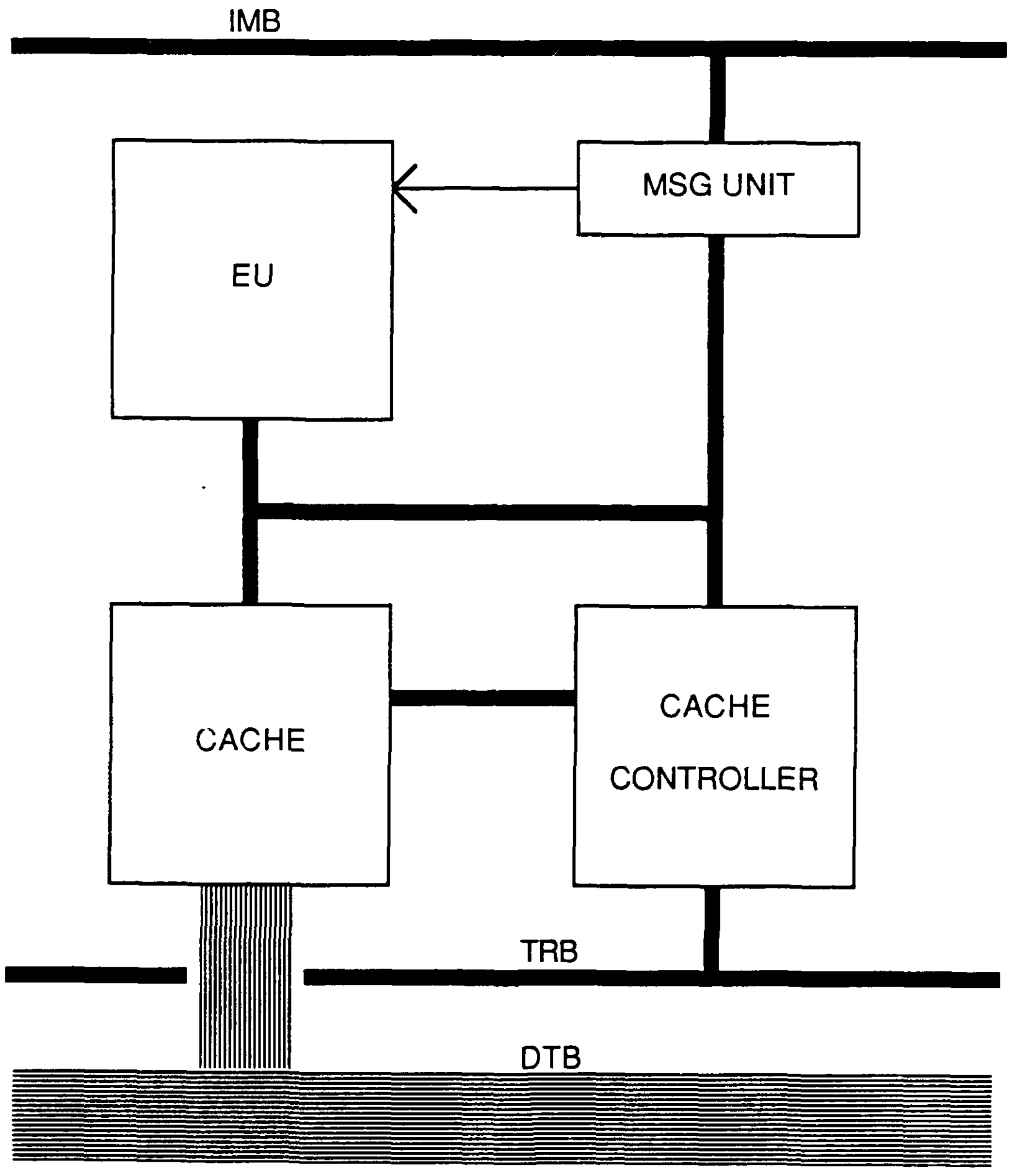


Figure 6: Prototype Processing Element

The prototype EU consists of a 68020 32-bit microprocessor with a 68882 floating point coprocessor, along with an EPROM which contains startup vectors and code, and some operating system kernel routines. The cache in the prototype PE will be 64 Kbytes in size, 2-way set associative. The cache controller will initially perform cache fetches only on misses, and will employ a write-back policy for cache writes. A more advanced cache controller which executes a variety of prefetching algorithms will be implemented in the future.

Not shown in the figure are a small amount of local RAM used by resident system software, and a simple access control circuit which ensures the isolation of process address spaces by trapping illegal access attempts.

VII. PROTOTYPE OPERATING SYSTEM

The operating system implemented on the prototype virtual port memory machine will be based on the system described in [JOHNS87], and will probably use portions of a multitasking kernel already implemented for 68000-family processors.

The major functions of this operating system will, of course, be similar to those of other multiprocessor operating systems; virtual memory management will reflect the virtual port memory environment, and will employ the page copy hardware for high-performance copy-on-write fault handling. Run-time linking may be accomplished as described in [JOHNS87].

This prototype will provide a test bed for evaluation of various processor allocation, memory management, and cache control strategies.

REFERENCES

JOHNS87	E.E. Johnson, *The Virtual Port Memory Multiprocessor Architecture*, PhD dissertation, New Mexico State University, 1987.
JOHNS88	E.E. Johnson, "The Virtual Port Memory Multiprocessor Architecture," Technical Report NMSU-ECE-88-001, 1988.
MOT85	Motorola, *VMEbus Specification Manual*, Revision C.1, October 1985.